NISTIR 88-3874

IMPLEMENTATION OF THE COORDINATE MEASURING MACHINE CONTROLLER

October 13, 1988

By:
Howard T. Moncarz Theodore H. Hopp Patrick Lezark

AMRF

National Institute of Standards and Technology
formerly

U.S. DEPARTMENT OF COMMERCE National Bureau of Standards Gaithersburg, Maryland

IMPLEMENTATION OF THE COORDINATE MEASURING MACHINE CONTROLLER

Howard T. Moncarz
Theodore H. Hopp
Patrick Lezark

October 13, 1988

Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

TABLE OF CONTENTS

		Page
I.	INTRODUCTION	1
	1. WHAT THIS DOCUMENT IS ABOUT	1
	2. AUDIENCE	1
	3. OVERVIEW	1
II.	TOP LEVEL DESCRIPTION OF THE CMM CONTROLLER	3
	1. DESCRIPTION OF THE CMM	3
	2. LOCATION IN THE WORKSTATION ARCHITECTURE	3
	3. MAIN CONTROLLER FUNCTIONS	3
	4. WORK ELEMENTS AND STATUSES	5
III.	INSPECTING PART GEOMETRY USING A CMM	7
	1. WHAT IS A FEATURE?	7
	2. WHAT IS A DIMENSIONAL TOLERANCE?	7
	3. TOLERANCES IMPLEMENTED	8
	3.1. DiameterTol	8
	3.2. FlatnessTol	8
	3.3. LengthTol	8
	3.4. AngleTol	9
	4. LOCATING THE PART	9
IV.	DATA STRUCTURES	11
	1. LOCAL DATA	11
	2. AMRF DATA	11

Page

V. TASK DECOMPOSITION 13

1. wsc_cmm 13

2. cmm_tol 13

3. tolerance 15

4. features 15

5. surfaces 16

6. points 16

7. machine 16

VI. PROCEDURE MODULES 19

1. cmm_lib 19

2. cmm_types 19

3. cm_glob 19

4. cmm_funcs 19

5. get_data 19

6. fitmod 19

7. comptrans 20

8. flb_lib 20
8.1. flb_fns 20
8.2. flb_errs 20
8.3. tpstuff 20
8.4. flb_talk 20
8.5. flb_init 21
8.6. flb_scan 21
8.7. report 21
8.8. flb 21

VII. INTERFACE TO EQUIPMENT 23

1. MODULES THAT INTERFACE TO EQUIPMENT 23

2. DETAILS OF THE CURRENT IMPLEMENTATION 24

3. CHANGES REQUIRED FOR EQUIPMENT SUBSTITUTION 25

Page

VIII. INITIALIZATION AND SHUT DOWN..... 27

1. START UP..... 27

2. SHUT DOWN..... 27

3. ABORT..... 27

IX. ERROR HANDLING..... 29

1. MISSED PART..... 29
1.1. Description..... 29
1.2. How Handled..... 29

2. UNEXPECTED TOUCH..... 29
2.1. Description..... 29
2.2. How Handled..... 29

3. PROGRAM HANGS..... 29
3.1. Description..... 29
3.2. How Handled..... 30

X. USER INTERFACE..... 31

1. STAND-ALONE OPERATION..... 31

2. USER COMMANDS..... 31

XI. FUTURE PLANS..... 33

1. SOFTWARE DEVELOPMENT..... 33

2. NEW HARDWARE..... 33

3. PROBLEM AREAS..... 34
3.1. Limitations On Parts That Can Be Inspected..... 34
3.2. Error Handling..... 34

APPENDICES

		Page
A.	IWS DOCUMENTATION LIST	35
B.	REFERENCES	37
C.	GLOSSARY (and abbreviations)	39
D.	FLAT FILE SPECIFICATIONS	41
	READER COMMENT FORM	53

LIST OF FIGURES

Page

Figure 1. Logical Architecture of the IWS..... 4

Figure 2. Control Levels for the CMM Controller..... 14

IMPLEMENTATION OF THE COORDINATE MEASURING MACHINE CONTROLLER

I. INTRODUCTION

1. WHAT THIS DOCUMENT IS ABOUT

This document describes the implementation specifics of the Coordinate Measuring Machine Controller (CMMC) program as of March 26, 1987. This program runs under the control of the ECS program that is described in the document Implementation of the Execution Control System of the Inspection Workstation [A.2]. The controller program consists of state machine modules that "customize" the controller for its particular application--i.e. supervising the CMM.

2. AUDIENCE

Anyone who needs to understand the internals of the CMM software should read this document. This includes anyone who will continue the development of the CMM software or make modifications to it.

The document Architecture and Principles of the Inspection Workstation [A.1] describes the principles that the ECS program and the CMM Controller program utilize. It is recommended that that document be read first.

3. OVERVIEW

Chapter II gives a top level description of the CMM Controller. It describes the equipment, specifies the location of the CMMC in the IWS control hierarchy, and describes the main functions the controller performs.

Chapter III discusses some general principles of dimensional inspection that the CMMC utilizes.

Next, Chapter IV describes the main data structures, both global to the AMRF as well as local to the IWS, that the controller program uses. The specific task decomposition that the CMMC incorporates is explained next in Chapter V. Additionally, procedure modules used by the main tasks discussed in Chapter V are described in Chapter VI.

The actual interface to the CMM is specified in Chapter VII. Specific details used in the start up and shut down procedures are described in Chapter VIII. Errors that can occur during operation are listed and explained in Chapter IX. Chapter X describes the user interface to the CMMC.

Finally, Chapter XI discusses future development plans for the CMMC.

The appendices include further information and implementation details. Appendix A lists the entire IWS documentation set. Other references are listed in Appendix B. Appendix C contains a glossary of terms used in this document. Appendix D specifies the internal file formats used to contain all the data used for a specific inspection.

Completing the document is a reader/comment form. You are encouraged to write down your comments and mail the attached form to the address specified.

II. TOP LEVEL DESCRIPTION OF THE CMM CONTROLLER

The CMM inspects dimensional tolerances of parts, and is supervised by the CMM Controller (CMMC).

1. DESCRIPTION OF THE CMM

The CMM used at the IWS is known as a horizontal arm CMM, since it consists of a horizontal arm that moves vertically. The part to be measured is placed on the table of the CMM. This table has three degrees of freedom in the horizontal plane--two translational degrees of freedom perpendicular to each other and one rotational degree of freedom about an axis through the center of the table. A probe is mounted on the end of the horizontal arm. The particular probe used for the IWS is a Renishaw PH9 type probe [B.2]. This probe has an additional two rotational degrees of freedom about the arm.

The probe is a force-sensitive mechanism that registers a touch if its synthetic ruby tip encounters a small force. This force is adjustable and is held at about five grams for normal IWS operation. At this force level, most parts can be accurately measured while the part is secured on the table by its weight. If the part is very light, it can still be measured if it can be mounted on a fixture, and the combined fixture and part placed on the table.

2. LOCATION IN THE WORKSTATION ARCHITECTURE

As shown in Figure 1, the CMM Controller is subordinate to the Workstation Controller (WSC) and is the supervisor to the CMM. In a future implementation, the CMM Controller will be able to access the IMDAS (Integrated Manufacturing Data Administration System), the distributed data system which provides common interfaces to the AMRF's user programs and underlying databases [B.6, B.10].

3. MAIN CONTROLLER FUNCTIONS

The main functions of the CMM Controller are the following:

Respond to WSC commands and return status information.

Retrieve inspection programs from the IMDAS.

Control CMM motion.

Analyze CMM data.

Report inspection results back to the AMRF.

Print local inspection reports.

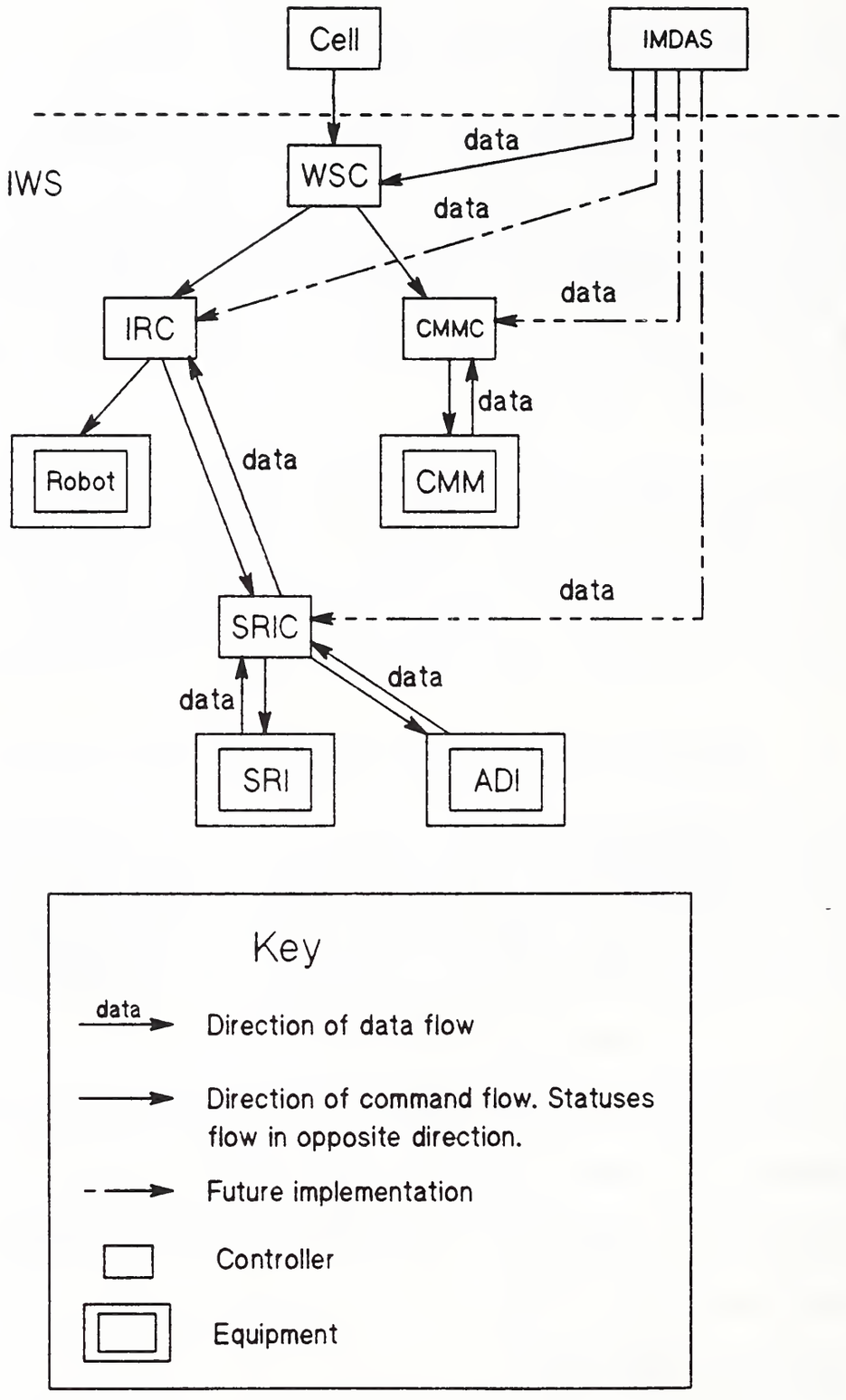

Figure 1. Logical Architecture of the IWS

4. WORK ELEMENTS AND STATUSES

The CMM Controller receives commands from the Workstation Controller. These commands are either transition commands (involved in the start up/shut down protocol), or work order commands (for operating in ready state) which contain the work elements that command the main functions that the CMM Controller is responsible for doing.

The work elements executed by the CMM Controller are:

LOAD_PART (loads the data for a part inspection)

INSPECT (directs the part inspection)

Statuses returned by the CMM Controller to the Workstation Controller are:

WORKING

DONE

ERROR

III. INSPECTING PART GEOMETRY USING A CMM

1. WHAT IS A FEATURE?

A part feature is a collection of part entities that is a useful grouping to satisfy a particular function. A geometric feature is a collection of part surfaces that are grouped together as a single entity. For the purpose of inspection, the entity chosen is one that can be measured. This is known as an inspectable feature, as contrasted by a machinable feature (which is a useful entity for machining). A feature can be machinable as well as inspectable, but that does not have to be the case.

In the IWS implementation, a feature is termed either simple or complex. A simple feature is represented by a single part surface. An example is that a hole can be represented by a single surface--a cylinder. A complex feature is a collection of simple features. An example is a pattern of holes. Each hole is a simple feature. The collection, taken as a single entity, is a complex feature.

Because of the categorization just described, measuring a feature always decomposes finally to measuring a surface. This entails measuring the location of a number of points on the surface.

The type of surface measured must be specified. In the IWS implementation the surface type can be any one of the following: a flat plane, a cylinder, a sphere, or a cone. A best fit of the points measured is determined for the surface type specified. This best fit is then used for tolerance determination.

2. WHAT IS A DIMENSIONAL TOLERANCE?

Dimensional tolerances are specified for a part to insure that a part manufactured to within those tolerances will serve its intended function.

As mentioned in section 1 above, a tolerance is referenced to a feature. There are two classifications for tolerances--intrinsic and extrinsic. An intrinsic tolerance depends only the feature being toleranced. On the other hand, an extrinsic tolerance is the measurement of a feature in relation to another feature (or set of features).

For example, to determine whether the diameter of a hole is within tolerance, only the hole needs to be measured to determine its diameter. This is a size tolerance and is classified as an intrinsic tolerance. On the other hand, the position of that same

hole is determined in relation to, for example, two other features--two sides of the part (each a flat plane). A position tolerance is an extrinsic tolerance.

3. TOLERANCES IMPLEMENTED

The intrinsic tolerances implemented for the IWS are diameter and flatness. The extrinsic tolerances implemented are length and angle. These are implemented as separate procedures. They are discussed below, using the same procedure names as are used in the program.

3.1. DiameterTol

This procedure determines whether the diameter of a hole or cylinder is in tolerance. The procedure compares the measured diameter with the design diameter and checks the difference against the tolerance specified for the hole or cylinder.

Diameter is an intrinsic tolerance.

3.2. FlatnessTol

This procedure determines whether a plane is within a specified flatness tolerance. It computes the distance from each measured point to the computed best fit plane, and it adds the distances of the two points which are farthest from the plane on each side. This sum is compared with the specified tolerance to determine if the plane is in tolerance.

Flatness is an intrinsic tolerance.

3.3. LengthTol

LengthTol checks several kinds of features: the distance from a feature plane, hole, or cylinder to a datum plane and the distance from a point (as determined by the intersection of three planes) to a datum plane.

In finding the distance from a feature plane to a datum plane, it is assumed that the two planes are parallel. LengthTol finds the centroid of the measured points of the feature plane and computes the distance from the centroid to the measured datum plane. This distance is compared with the specified dimension and tolerance.

For a hole (or cylinder) whose axis is supposed to be parallel to the datum plane, LengthTol finds the distance from an arbitrary point on the axis of the hole or cylinder to the measured datum plane. The coordinates of this arbitrary axis point are computed by the module "fitmod" as parameters of the surface.

To find the distance from a point (such as a corner) to a datum plane, LengthTol first finds the coordinates of the point of intersection of the three planes. LengthTol computes the distance from this point to the datum plane.

A feature is considered simple if it can be represented by a single surface. A feature is referred to as a pattern if more than one surface is needed to represent it. For example, single planes, holes, and cylinders are simple features, while a point is a pattern because it is actually represented by more than one surface.

These tolerances are all extrinsic.

3.4. AngleTol

AngleTol finds the angle between two planes and compares this to the toleranced dimension. The angle between two planes is computed by finding the angle between the normals to the planes. Since the normals have length 1, the angle is given by the arccosine of the dot product of the two normal vectors. This computed angle is compared with the design angle and the specified tolerance.

Angularity is an extrinsic tolerance. Of the two planes that determine the given angle, one is treated as a datum plane, the other as the feature.

4. LOCATING THE PART

The IWS robot places the part on the CMM table at approximately the location specified by data. However, in order for the CMM to probe the correct features of the part without getting an accidental touch at a different feature or missing the feature altogether, it is necessary that the CMM Controller know the part location more precisely. To accomplish this, the CMM Controller directs the CMM to search for the part before inspecting it. Four points are probed--three on the part and one on the CMM table. Based on these measurements, a coordinate frame referenced to the part's main axes is determined and used throughout the rest of the inspection.

IV. DATA STRUCTURES

1. LOCAL DATA

In the March, 1987 implementation, the data required for a dimensional inspection of a part is stored locally in disk files on the CMMC itself. The CMMC is sent the command to 'LOAD_PART' to tell it what part needs to be inspected. The LOAD_PART command is packaged with two arguments--the part name and the name of the inspection plan (the latter is an integer). Upon receiving this command the CMMC prepares to access the proper data during the inspection.

Once the CMMC completes the command to 'LOAD_PART', it waits for the command to 'INSPECT'. After receiving the INSPECT command, it directs the inspection of the part. As each state machine module runs, any data required by that module is retrieved from the local data as it is required. It is necessary to access the data locally during the actual inspection to achieve the required response time.

The local data is stored in a flat file system--each relation is stored in a separate file. A relation contains key fields that are used to find the record required. Then, the data fields in that record are retrieved. The full specification for all the CMMC flat files is in Appendix D. The description of how the flat file system is implemented is in the IWS document Implementation of the Execution Control System of the Inspection Workstation [A.2].

2. AMRF DATA

In a future implementation the CMM Controller will retrieve the data it requires to inspect a particular part from the AMRF IMDAS. This data will be contained in two separate files. The first file is the part model file. The format for this file has been specified [B.8], and part model data for a number of parts already resides in the AMRF IMDAS. Any process connected to the AMRF network can retrieve this data. The part model file contains the geometry, topology, and tolerance data for a part in a neutral format that can be used by all processes throughout the AMRF.

Supplementary to the part model data, additional data is required to inspect a part. This data includes sequence information (i.e. which tolerance to inspect next), probe data for each point to be inspected (i.e. which PH9 tip orientation to use), etc. All of this data will be contained in a single file, the inspection data file, which has a format similar to the part model file. The current implementation does not use this file, and it has not yet been specified.

After the two files are retrieved, they will be transferred to local data files (described in the previous section) that allow quick retrieval of individual pieces of data as the controller program needs them. The structure for these local data files will remain the same, even when the data originates from the IMDAS rather than from the CMMC itself (as is currently the case).

V. TASK DECOMPOSITION

The state machine modules used to implement the CMM Controller are shown in Figure 2. Listed in their order of hierarchical task decomposition (from highest to lowest), those modules are wsc_cmm, cmm_tol, tolerance, features, surfaces, and points. The lowest level module in the CMM Controller task decomposition is the machine module.

The machine module is implemented as a procedure module rather than as a state machine module. This has the advantage of reducing the number of state machine cycles necessary to execute a command, and therefore increasing the speed. However, the disadvantage is that the ability to react to feedback from the equipment at the machine module level is more difficult. That feedback must be passed from the machine module's supervisor down to the machine module by a procedure call.

This section lists each of the state machine modules, including the machine module, and provides a general description of each.

1. wsc_cmm

The "wsc_cmm" module implements an AMRF standard control protocol, called the UVA Protocol, [B.3], in interfacing the CMM Controller to the Workstation Controller. This module accepts commands from the WSC and passes those commands down the task decomposition hierarchy. It receives statuses from its subordinate, the cmm_tol module, and returns statuses to the WSC.

2. cmm_tol

In the generic controller model, "cmm_tol" is the main task module. This module supervises the main functions performed by the CMM Controller, namely, to retrieve the data to inspect a part (given the part name and the inspection plan name) and to supervise the inspection. A separate work order is issued to cmm_tol for each of these functions--'LOAD_PART' and 'INSPECT', respectively.

Currently, cmm_tol retrieves data from data files residing locally on the CMM Controller. Eventually, this data will be retrieved from the IMDAS. Once retrieved, this data will be parsed and stored in local data files in the same format as the data that is currently being used.

Included in the data that cmm_tol retrieves is data that specifies the approximate location of the part on the CMM. The cmm_tol module uses that data to determine the part coordinate system, which will be used when the CMM Controller locates the part.

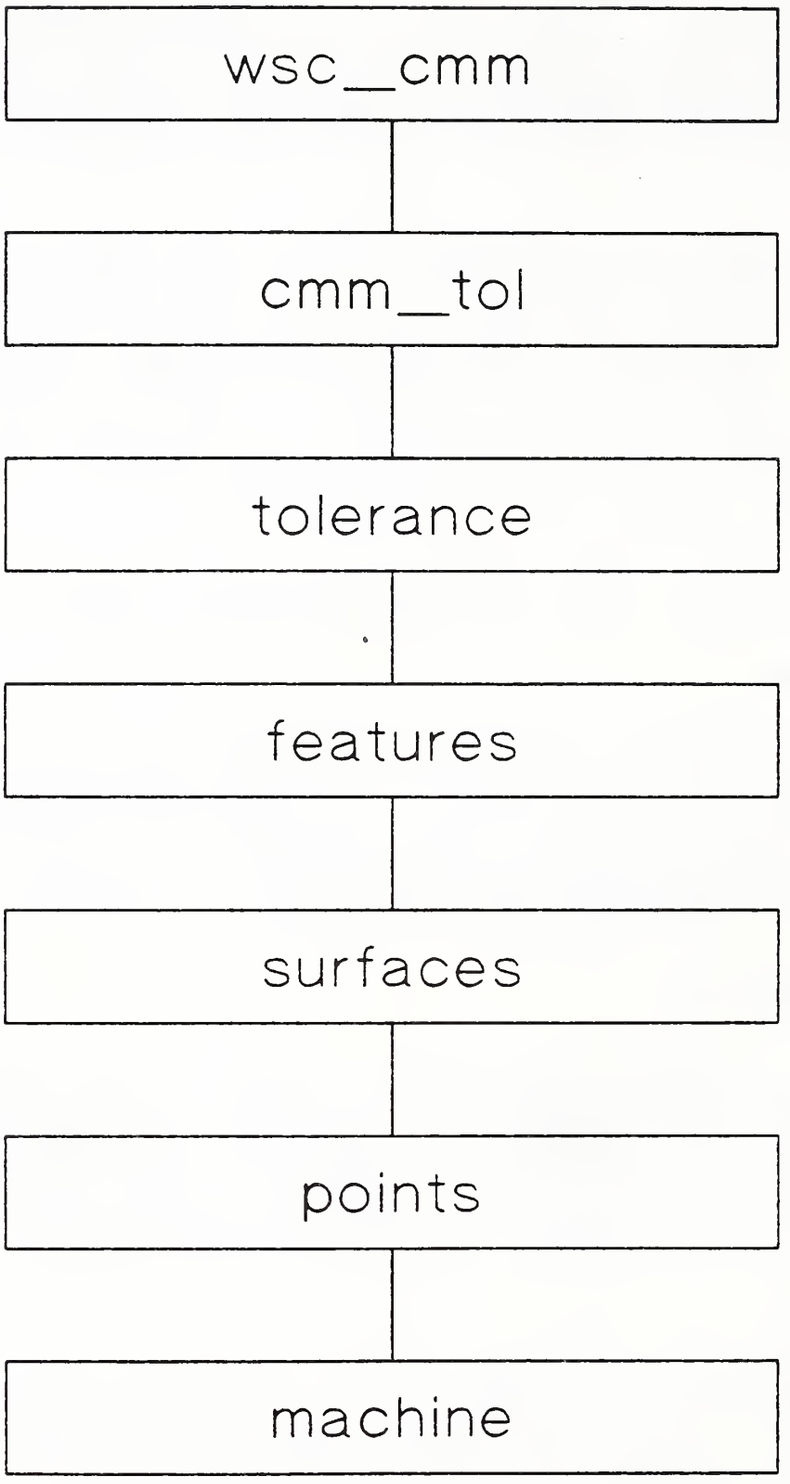

Figure 2. Control Levels for the CMM Controller

After cmm_tol retrieves the data and completes the determination of the part coordinate system, cmm_tol reports a status of 'DONE' to wsc_cmm, indicating that the order 'LOAD_PART' is completed.

When cmm_tol gets the order to 'INSPECT', it first directs the CMM to go to a safe point, specified by data. Then it directs the CMM to locate the part, using the coordinate system determined during the execution of the 'LOAD_PART' command. After locating the part, the part coordinate system is recalculated to reflect the improved measurements just made.

Using this new coordinate system, the part is inspected. The CMM is then directed to return to the safe point and wait for the next command. The 'INSPECT' command is completed, and cmm_tol reports a status of 'DONE' back to wsc_cmm.

3. tolerance

The "tolerance" module takes charge of directing the inspection for a given tolerance. The main command executed by this module is 'INSPECT_TOLERANCE', issued from cmm_tol. The command includes the tolerance name, a key into the data.

Upon receiving this command, the tolerance module looks up the data describing the tolerance required. This data specifies which feature is being toleranced and whether the tolerance is intrinsic or extrinsic. If intrinsic, a command is issued to the subordinate module, features, to measure the feature being toleranced. If the tolerance is extrinsic, a command is issued to "features" to measure the feature (or features) that represent the data reference frame (DRF). After the DRF is measured (and recorded), a command is issued to "features" to measure the feature being toleranced. Once the feature is measured (whether the tolerance is intrinsic or extrinsic), it is recorded and the tolerance is computed. The command is completed and the module tolerance reports 'DONE' back to cmm_tol.

4. features

The "features" module measures the features specified by the tolerance module. If "features" is commanded to measure a data reference frame, "features" measures all the features that determine that DRF. Also, "features" must determine whether each feature it is to measure is simple or complex. If the feature is simple, then a command is issued to the module, surfaces, to measure the surface representing that feature. Otherwise, if the feature is complex, "features" must send individual commands to the surfaces module to measure each surface that combine to make up the complex feature. After each surface is measured,

"features" records the results for that surface. When the entire feature is measured, "features" reports 'DONE' back to the tolerance module.

5. surfaces

When "surfaces" gets the command 'MEASURE', it looks up the surface to be measured and gets the information that describes that surface. That information includes the type of surface (flat plane, cylinder, etc.), the coefficients that define that surface analytically, the number of points to be measured on that surface, the probe tip diameter, etc. The module then issues commands to the points module to measure the points that are needed to determine that surface. Each point is recorded as it is measured. When all points required for the surface are recorded, a best fit for the surface is computed and also recorded. The command is completed and 'DONE' is reported to "features".

6. points

The main duty of the "points" module is to direct the CMM to go to the next point commanded by the surfaces module. That point can be either an inspection point or a point in space, referred to as a safe point. It must do this without colliding with the part in going from the current point to the next one directed. Also, this module is responsible for directing the approach of the CMM's probe to an inspection point from the proper offset to that point and at the correct speed. It then directs the CMM to retract its probe to the correct offset from the inspection point after touching the part.

Each inspection point has a collection of data associated with it that allows the points module to accomplish its task. This data is divided into data specifying point information and data describing characteristics of the CMM probe. The point information includes the coordinates of the point, the orientation and offset from which to approach that point, and path information to the approach point. The probe characteristics include the angle of the tip, the free space speed up to the approach point, the approach speed, the retract distance, and the overtravel distance. The overtravel distance specifies the distance the probe tip should move beyond the specified inspection point before that point is considered "missed".

7. machine

This module is the bottom level task in the CMM Controller task decomposition. It packages up commands (that are independent of a specific CMM--i.e. logical CMM commands) from other tasks and interfaces the CMM Controller to the CMM (a specific CMM) by

referencing the flb module. As mentioned above, the machine module is implemented as a set of procedures rather than as a state machine.

The CMM at the IWS contains proprietary firmware that interfaces to the actual servos, encoders, etc. Provided along with this software is a program, written in BASIC, which runs on an HP 9836 computer that interfaces to the proprietary firmware. The BASIC program is called flb, which stands for function library. This program has been translated to a Pascal module, and is considered part of the CMM, rather than the CMM Controller program.

VI. PROCEDURE MODULES

The modules in the task decomposition (state machine modules plus the module machine) use procedures that are packaged into separate modules. These modules are described in this section.

1. cmm_lib

This library module contains procedures from the HP library that are not used by the other controllers. The particular modules included here are used to interface the HP computer to the CMM using the HPIB I/O board.

2. cmm_types

This module includes data structure types that are specific to the CMM Controller, and are referenced by state machine modules as well as procedure modules throughout the CMM Controller program.

3. cm_glob

This module is used to communicate a couple of large variables between the tolerance and features modules, rather than through common memory. This is done because the current implementation of common memory transfers all common memory variables every cycle. The variables included in cm_glob are needed very infrequently, and since they are so large, it would slow up the whole controller if these variables are transferred every cycle.

4. cmm_funcs

This module contains some general functions that are specific to the CMM Controller and are referenced by modules throughout the CMM Controller program.

5. get_data

This module contains the procedures that search the locally stored data files to return the data values needed by the state machine modules as they are running.

6. fitmod

The module fitmod uses least squares models to fit sets of points to the parameters of several surfaces--plane, sphere, cylinder, cone, and torus.

7. comptrans

This module computes a transformation matrix which compensates for the approximations made in placing a part to be measured on the CMM.

8. flb_lib

The flb_lib module interfaces the CMM Controller to the CMM. This module is a rewrite of a BASIC program (supplied by Sheffield [B.1]) to Pascal. In the Pascal version, the program is divided into separate modules that are linked together into the flb_lib module. These modules are described in this section. All of the capabilities provided in flb_lib can be accessed by the CMM Controller program. However, only a small portion of these are utilized when operating in the normal IWS configuration. If the capabilities are not normally specified, it is noted in the description for that module.

8.1. flb_fns

This module contains procedures that are used by many of the flb_lib modules, and are specific to the CMM Controller program.

8.2. flb_errs

This module handles errors detected by the CMM in the manner specified by the Sheffield protocol. It contains one procedure that allows the user to substitute the user's own error handling procedures in place of that protocol. This, in fact, is used in the CMM Controller program.

8.3. tpstuff

The module tpstuff contains procedures that compute tolerances based on different material conditions. The current CMM Controller implementation does not use these.

8.4. flb_talk

The protocol specified by Sheffield that handles the communication between the HP computer and the CMM is quite complicated, and is implemented in this module. It is the only module in the CMM Controller program that interfaces directly to the CMM.

8.5. flb_init

The procedures contained in flb_init are used to cold start or warm start the CMM. The CMM Controller uses the cold start procedure.

8.6. flb_scan

This module provides procedures to take multiple probe readings while the probe touches multiple points on the surface of a part by issuing one command. This module is not used by the CMM Controller.

8.7. report

This module allows the user to specify the format for a printout of the results of an inspection, and to actually perform that printout. This module is not used by the CMM Controller.

8.8. flb

This module provides the interface between the CMM Controller program and the flb_lib module.

VII. INTERFACE TO EQUIPMENT

One goal of the IWS project is to design the software so that higher levels of control are device independent. Then, any piece of equipment can be interfaced to the system by writing a device interface.

This is analogous to a graphics system, where higher level modules in the system refer to logical devices. A logical device is a module that performs a particular function required by the system. In its common usage, a logical device is the lowest level module in the system that is still device independent. By writing a module that interfaces an actual physical device to the logical device, that physical device can be interfaced to the system without changing the system. This design gives the system more generality and flexibility.

For example, a locator device is a logical graphics input device. The actual hardware to perform this function can be chosen from a collection of different types of devices, including digitizing tablets, light pens, mouses, etc. The light pen would perform this function most directly, but other types of devices, such as digitizing tablets, can simulate the function through software. In other words, the hardware that interfaces to the system cannot only be from different manufacturers, but it can even be a different type altogether, as long as it can perform the function (or functions) specified by the logical device.

This is the perspective from which the IWS controllers were designed. For the CMM Controller, the "machine" module is the device interface to the CMM. (Note that the flb module is considered part of the CMM--see Chapter V, Section 7.) The module, points, could be considered the logical device. It defines the lowest level procedures that are still device independent. Thus, by changing the machine module, a CMM from a different manufacturer could be interfaced to the CMM Controller.

In the current design, this clear division has not been strictly upheld. This will be corrected in the future. However, for now the interface between the CMM Controller and the CMM is more complicated, and the details of that interface are explained below.

1. MODULES THAT INTERFACE TO EQUIPMENT

The modules that interface to the CMM are machine, points, and cmm_tol. The CMM functions accessed by these modules are described in the next section.

2. DETAILS OF THE CURRENT IMPLEMENTATION

In the machine module, the following CMM functions are performed:

Initialize the CMM from a cold start, and store the initial machine reference frame for recall upon command.

Move the probe to a specified location in free space.

Move the probe and touch the part at a specified location on the part, and save the location data for the point touched.

In the points module, the following CMM functions are performed:

Set the probe free space speed.

Set the probe touch speed.

Set the probe retract distance (after a touch).

Set the probe overtravel distance (before reporting a miss).

Select which PH9 probe tip orientation to use.

In the cmm_tol module, the following CMM functions are performed:

Set the units of measurement (either inches or millimeters).

Set the probe free space speed (also set in module points).

Establish the part reference frame given the 4x4 transformation of machine to part coordinates, and store that reference frame for recall upon command.

Recall the machine reference frame.

Recall the part reference frame.

Set the CMM method of error handling to allow the program on the HP to handle it, rather than allow the CMM firmware to handle it in its default specification.

Move the probe to a specified location in free space. (This is currently implemented by referencing the machine module function to accomplish this.)

3. CHANGES REQUIRED FOR EQUIPMENT SUBSTITUTION

To substitute the CMM from a different manufacturer, the functions specified in section 2 above must be implemented for that CMM. For the particular CMM currently implemented, some higher level functions (for example, setting the part reference frame) are implemented by the CMM itself. If that is not the case with the new CMM, those functions must be implemented in software.

VIII. INITIALIZATION AND SHUT DOWN

The top level module of the CMM Controller, wsc_cmm, implements the UVA initialization and shut down protocol [B.3].

This section discusses what is done in the CMM Controller when it receives these transition commands. It ignores the interface between wsc_cmm and the Workstation Controller, since this is fully covered in the AMRF document mentioned above.

1. START UP

On cold start up, i.e. when the CMM Controller program is first started, the CMM is sent a command to initialize the CMM. Additionally, common memory mailboxes are established. After these events occur, the CMM Controller is ready to receive commands. When the 'WARM_STARTUP' command is received, cmm_tol goes into its 'IDLE' state, reports 'DONE' back to wsc_cmm, and is then ready to receive order action commands.

2. SHUT DOWN

On receiving the 'WARM_SHUTDOWN' command, the CMM is commanded to return the probe to its initial power up position--at the gage block. Then cmm_tol reports the status 'DONE' to wsc_cmm.

3. ABORT

The ABORT command is used for error handling. Currently, this command is not implemented. When the command is passed down from wsc_cmm to the next level, cmm_tol, it is ignored.

IX. ERROR HANDLING

This section discusses the types of errors that can occur during the CMM Controller operation. Currently, any error crashes the program. The three main errors that have been observed are discussed below.

1. MISSED PART

1.1. Description

This error occurs when the CMM probe is directed to touch the part at a certain location, and it does not contact the part at that location. Before getting this error, the probe will search a distance past the specified location equal to the overtravel distance. This error can occur when the CMM Controller is trying to locate the part or when it is inspecting the part.

1.2. How Handled

The CMM itself detects this error and kicks over into MANUAL mode. The CMM Controller currently cannot recover from this error.

2. UNEXPECTED TOUCH

2.1. Description

This error occurs when the CMM probe contacts the part, or anything else, when it has not expected a touch. This error can occur when the CMM Controller is trying to locate the part or when it is inspecting the part.

2.2. How Handled

The CMM itself detects this error and kicks over into MANUAL mode. The CMM Controller currently cannot recover from this error.

3. PROGRAM HANGS

3.1. Description

This error has three main sources. First of all, when the Controller is communicating with the CMM, if the handshaking between the two does not conform to the proper protocol, the Controller program can hang. It will continually wait for a certain response from the CMM that will never occur.

Secondly, it is possible that the surface fit or tolerance algorithms will not converge. This can cause the program to go into an infinite loop.

Finally, any bug in the control structure of the program can force the Controller to go into an undetermined domain and effectively crash the program.

3.2. How Handled

Currently, the CMM Controller cannot recover from this error.

X. USER INTERFACE

1. STAND-ALONE OPERATION

In integrated mode, wsc_cmm is the highest level module in the CMM Controller. It receives commands, via the local network, from the Workstation Controller. If the CMM is to be run in stand-alone mode, the operator needs to enter commands to the controller and have those commands transferred to the wsc_cmm module. The interface that provides this connection between the user and the wsc_cmm module is the cmmtest module. Cmmtest simulates the WSC, and allows the user to select commands for the CMM Controller.

2. USER COMMANDS

The commands available to the user are 'ABORT', 'SHUTDOWN', 'STARTUP', and 'EXECUTE'. These are the transition commands. The first three commands do not have arguments. The arguments for 'EXECUTE' are the work orders and their respective arguments.

The user must first choose 'STARTUP' to transfer the CMM Controller to the ready state. Next, the user selects 'EXECUTE' and may select either of the two work orders: 'LOAD PART' or 'INSPECT'. 'LOAD PART' is used to select the data to inspect a particular part. 'INSPECT' directs the CMM Controller to start the inspection using the data just loaded by the 'Load Part' work order.

After a part has been inspected, data for a new part may be loaded and inspected, or else the CMM Controller may be shut down by issuing the transition command 'SHUTDOWN'.

XI. FUTURE PLANS

1. SOFTWARE DEVELOPMENT

Software development is continuing in a number of areas. The primary effort is involved with fully integrating the CMM into the AMRF. This entails retrieving the data required to inspect a part from the IMDAS, and translating that data to the internal data structures required by the CMM Controller.

Another area of work which would be useful to do would be to extend the types of tolerances that can be measured. In fact, the entire subject of relating CMM data to functional tolerances needs detailed research in concert with comprehensive analysis of actual data.

Currently, the path that the probe tip of the CMM takes to go from one point to another is specified by data that is determined ahead of time. Research is proceeding that will consider whether the CMM Controller can determine the correct path in real time. Hopefully some method can be implemented that will do automatic collision avoidance quickly enough to do it on-line, rather than in an off-line programming system.

Automatic collision avoidance would be necessary to proceed to another interesting area of research--adaptive CMM inspection. During an actual part inspection, an analysis of the data as it is collected might be useful for determining what additional measurements are necessary for completing the analysis.

2. NEW HARDWARE

No plans for new hardware are immediately eminent. However, two hardware components have been identified that would be very useful additions to the CMM. An automatic probe changer would allow the CMM to inspect a wider variety of parts than it currently is able to do while integrated into the IWS. Also, more surfaces of certain parts could be inspected without refixturing.

Secondly, an automatic vise, mounted on the CMM, would allow the CMM to inspect light parts that cannot be secured by gravity alone. Additionally, parts that cannot lie in a stable position on the CMM's table (i.e. a cylindrical part) could be secured by the vice as well.

3. PROBLEM AREAS

3.1. Limitations On Parts That Can Be Inspected

Before inspecting the part, the CMM Controller goes through an algorithm to locate that part. Currently, the location algorithm requires that the part have two flat surfaces that are perpendicular to each other, are perpendicular to the table of the CMM, and are parallel to two of the CMM's axes. A part that does not have two such planes (i.e. a cylinder with no flat planes) cannot be measured.

A part that is so light that it would be disturbed by the force of the probe touching it cannot be measured, unless it is secured by a fixture. For the IWS to handle it, the part would have to be delivered with the fixture already holding it.

Along the same lines, a part that does not have a flat surface to lie on while it is being probed, cannot be inspected unless it is delivered to the CMM already secured by a fixture.

3.2. Error Handling

Error handling has not been implemented. Currently, any error, such as an unexpected probe touch, will crash the system. This is an area that is under current investigation. For the near term, error handling procedures--preferably automatic, but including manual intervention--are being developed for specific types of errors. In the longer term, we are looking into developing general error handling procedures that would be suitable for any controller.

One error that will cause the CMM probe to get an unexpected touch is not easily resolved. If the robot puts the part on the table where it can be located by the CMM, but in a rotated position, it is possible that the CMM probe cannot reach a feature that it could reach if the part were in the correct position. For example, probing into a small diameter horizontal hole could cause the probe to hit the edge of the entrance to the hole rather than the surface of the hole itself.

APPENDICES

A. IWS DOCUMENTATION LIST

1. H. T. Moncarz, Architecture and Principles of the Inspection Workstation, NBSIR 88-3802, June 9, 1988.

2. H. T. Moncarz, Implementation of the Execution Control System of the Inspection Workstation, NBSIR 88-3787, May 19, 1988.

3. H. T. Moncarz, T. H. Hopp, and P. Lezark, Implementation of the Coordinate Measuring Machine Controller, NISTIR 88-3874, October 13, 1988.

4. H. T. Moncarz and T. V. Vorburger, Implementation of the Surface Roughness Instrument Controller, NBSIR 88-3794, June 2, 1988.

5. H. T. Moncarz and B. Borchardt, Implementation of the Inspection Robot Controller, NBSIR 88-3772, April 21, 1988.

6. S. A. Osella, Implementation of the Inspection Workstation Controller, NBSIR 88-3819, July 13, 1988.

7. J. Zimmerman, Inventory of Equipment in the Inspection Workstation, NBSIR 88-3775, May 5, 1988.

8. H. T. Moncarz, S. A. Osella, B. Borchardt, and R. Veale, Operations Manual for the Inspection Workstation, NBSIR 88-3766, April 21, 1988.

9. J. Zimmerman, Recommended Technical Specifications for Procurement of Commercially Available Systems for the Inspection Workstation, NBSIR 88-3779, 1988.

B. REFERENCES

1. The Bendix Corporation, Cordax Coordinate Measuring System, Reference Manual for the CORDAX Function Library, October, 1982.

2. The Bendix Corporation, User's Manual for the PH9 Probe Head Expansion, May, 1982.

3. D. R. O'Halloran and P. F. Reynolds, Jr., "A Model for AMRF Initialization, Restart, Reconfiguration, and Shutdown", NBS/GCR 88/546, May 23, 1986.

4. R. Rybczynski, et al., AMRF Network Communications, NBSIR 88-3816, June 30, 1988.

5. C. E. Wenger, Material Handling Workstation Implementation, NBSIR 88-3784, May 19, 1988.

6. C. Furlani, et al., Integrated Manufacturing Data Administration System, (IMDAS) to be published as an NISTIR, 1988.

7. T. H. Hopp and K. C. Lau, "A Hierarchical Model-Based Control System for Inspection," Automated Manufacturing, ASTM STP 862, L. B. Gardner, Ed., American Society for Testing and Materials, Philadelphia, 1985, pp. 169-187.

8. T. H. Hopp, AMRF Database Report Format: Part Model, NBSIR 87-3672.

9. C. R. McLean, Control Systems Command Feedback Protocol Document, to be published as an NISTIR.

10. D. Libes and E. Barkmeyer, "The integrated manufacturing data administration system (IMDAS)--an overview", International Journal of Computer Integrated Manufacturing, Vol. 1, No. 1, pp. 44-49.

C. GLOSSARY (and abbreviations)

CMM Abbreviation for the Coordinate Measuring Machine.

CMMC Abbreviation for the CMM Controller.

controller
Supervises the operation of a mechanism, another controller, or both.

coordinate measuring machine
Machine used to measure the dimensions of a part.

ECS Abbreviation for the execution control system.

extrinsic tolerance
Tolerance based on the measurement of a feature in relation to another feature.

execution control system
Computer program that runs on each controller computer and implements the AMRF design principles. This program loads and executes those modules which determine which controller is actually being run.

inspection workstation
AMRF workstation that inspects parts for dimensional tolerance and surface finish.

intrinsic tolerance
Tolerance that depends only on the feature being measured.

IWS Abbreviation for the Inspection Workstation.

logical architecture
Specifies the direction of commands and statuses between controllers and between controllers and equipment.

PH9 probe
A motorized probe used on the CMM. This probe can be oriented automatically under computer command. This allows the CMM to measure more part surfaces, without reorienting the part, than would be normally possible.

physical architecture
Specifies the physical connections among the controllers and equipment.

ready state
: The state in which a controller is ready to accept work order commands. This is the normal state of the controller during its operation.

state machine
: Software control unit with outputs dependent on inputs to it plus its internal state. This is the building block for the IWS control software.

transition commands
: Commands used to transfer the IWS to a new state (specified by the UVA protocol).

UVA Protocol
: Model, proposed by research group from the University of Virginia and adopted by the AMRF, that specifies the start up and shut down sequence for the AMRF as a whole as well as every controller within the AMRF.

work element
: The part of the work order command that specifies what main controller function to perform.

work order commands
: A command accepted by a controller when it is in ready state, and used to perform one of its main functions (specified by the work element).

WSC
: Abbreviation for the Workstation Controller.

D. FLAT FILE SPECIFICATIONS

This appendix contains the specifications for the flat files used by the CMM Controller and contained in local disk files. For details concerning the implementation of the flat file system, see the IWS document Implementation of the Execution Control System of the Inspection Workstation [A.2]. For a general description of what the flat files are used for, see Chapter IV of this document.

The format for specifying these files (referred to as relations) is described here. Each file is composed of ASCII characters that are broken up into records--each record containing one or more fields. Records are separated by a carriage return and a line feed. Fields are separated by one or more spaces.

The description for each relation begins with the name of the relation. The name given here is the same as given in the computer program, except that in the computer program the name is prefixed by 'DS_'.

A brief description of the relation is specified next.

This is followed by the name of an example data file that is actually used. The data files referenced contain data to inspect the pipe clamp, which is one of the parts commonly manufactured at the AMRF.

The task module from which this relation is retrieved is specified next.

This is followed by the names of the key fields. These fields are used to find the particular record in the relation that is required. The names of these fields often include the underscore character, so that a name will clearly specify a single field, even if it has more than one word. However, these names are not necessarily the same as they appear in the computer program.

Additionally, the data types for these fields are indicated by following the data field (or fields) by ' : ' and then the identification of the type. (Many of the fields are of type integer.) Furthermore, comments are often included that elaborate on what the fields mean. These are distinguished by enclosing them between braces, '{' and '}'.

Following the names for the key fields are the names for the data fields. The names are specified in the same manner as those for the key fields. The remarks above concerning the data type information and commenting apply here as well.

1. PathData

Description:
Specifies the path (integer name for it and the number of points on it) to follow based on the current location and the destination location.

Example file name: path_clp

Retrieved from: points

Key fields:
Current_Surface, Current_Point_Number,
Destination_Surface, Destination_Point_Number : integer

Data fields:
Path_Name, Path_Length : integer
{Note: include the safe point in this file -- point number for the safe point must be specified as 0. Surface number is whatever it was defined to be in the relation SafePos.}

2. ToolChange

Description:
Specifies the path to follow if the probe or probe tip needs to be changed plus the method of changing it. (Currently the tool change method is not implemented.)

Example file name: tpat_clp

Retrieved from: points

Key fields:
Current_Surface, Current_Point_Number, Current_Tip_Index,
Destination_Surface, Destination_Point_Number,
Destination_Tip_Index : integer

Data fields:
To_Path_Name, To_Path_Length,
Tool_Change_Method,
From_Path_Name, From_Path_Length : integer

3. PathStep

Description:
Specifies the point name based on the path name and index.

Example file name: ps_clp

Retrieved from: points

Key fields:
Path_Name, Path_Point_Index : integer

Data field:
Path_Point_Name : integer

4. PathPoint

Description:
Specifies the three components of the path point based on the point name.

Example file name: ppts_clp

Retrieved from: points

Key fields:
Path_Point_Name : integer

Data field:
three components of point (x, y, z) : real
{Note: do not include the safe point in this file.}

5. PointData

Description:
Specifies the three components of the point to be probed plus the three components of the direction from which to approach the point, based on the surface name and point number.

Example file name: pts_clp

Retrieved from: points

Key fields:
Surface_Name, Point_Index : integer

Data fields:
three components of point (x, y, z),
three components of approach vector : real

6. FeatrTyp

Description:
Specifies whether a particular feature is simple (S) or a pattern (P). If a pattern, the feature is composed of more than one surface.

Example file name: fdsc_clp

Retrieved from: features

Key field:
Feature_Name : integer

Data field:
Feature_Type : (S, P)

7. WhatSurf

Description:
Given the name of a simple (S) feature, return the surface name.

Example file name: simp_clp

Retrieved from: features

Key field:
Feature_Name : integer

Data field:
Surface_Name : integer

8. NoSubs

Description:
Given the feature name, return the number of sub-features it has. If it is a simple feature, the number returned will be one; else it is a pattern, and the number will be greater than one.

Example file name: patr_clp

Retrieved from: features

Key field:
Feature_Name : integer

Data field:
Number_of_Subfeatures : integer

9. WhatFeatr

Description:
Return the simple feature name given the index to a complex feature.

Example file name: subf_clp

Retrieved from: features

Key fields:
Complex_Feature_Name : integer {feature is a pattern}
Pattern_Index : integer {order to inspect features}

Data field:
Feature_Name : integer {for simple feature}

10. ProbeTipData

Description:
Specifies probe name, the tip number, and the tip effective radius, given the probe tip index.

Example file name: tips_clp

Retrieved from: points

Key fields:
Probe_Tip_Index : string

Data fields:
Probe_Name : identifier
Tip_Number : integer
Tip_Radius : real

11. NoDatums

Description:
Specify the number of datums (reference surfaces) given the data reference frame name.

Example file name: drf_clp

Retrieved from: features

Key fields:
DRF_Name : integer {data reference frame name}

Data field
Number_of_Datums : integer {for true position in 3-D space, require 3 datums}

12. DatumName

Description:
Return the feature name (for now it must be a simple feature) given the data reference frame name and index.

Example file name: datm_clp

Retrieved from: features

Key fields:
DRF_Name : integer {index -- ranges from 1 to Number_of_Datums}
DRF_Index : integer {This is actually the precedence -- i.e. specifies the primary, secondary, or tertiary axis.}
Data field:
Feature_Name : integer {must be a simple feature}

13. InTol

Description:
Assuming the tolerance is intrinsic, return the specific data required for that tolerance.

Example file name: intol_clp

Retrieved from: tolerance

Key fields:
Tolerance_Name : integer {this is the same Tolerance_Name as given in TolDescr}
Data field:
ToleranceCharacteristic : string {FLATNESS, DIAMETER, LOCATE}

```
        HiLim : real    {tolerance for upper limit, i.e. not the
                         upper limit itself.  If only 1 tolerance
                         value required, for example, FLATNESS, then
                         HiLim used for it.}
        LowLim : real {tolerance for lower limit}
                  {Note: this will be blank if only 1 tolerance
                  value required}
        Third : real    {leave blank if this is a tolerance -- i.e.
                         FLATNESS or DIAMETER, BUT if this is for
                         LOCATE (to locate part), then this real
                         number is the third component of the origin
                         translation (the HiLim and LowLim will
                         become the first 2 components of the origin
                         translation}
```

14. ExTol

Description:

Assuming the tolerance is extrinsic (i.e. given as EX in Tolerance_Type above), return the specific data required for that tolerance.

Example file name: xtol_clp

Retrieved from: tolerance

Key fields:

```
        Tolerance_Name : integer {this is the same Tolerance_Name
                                  as given in TolDescr}
```

Data field:

```
        ToleranceCharacteristic : string  {LENGTH, ANGLE}
        HiLim : real    {tolerance for upper limit, i.e. not the
                         upper limit itself}
        LowLim : real   {tolerance for lower limit}
                         {for ANGLE, give in degrees}
        DRF_Name : integer  {points to DRF required to measure this
                             tolerance}
```

15. TolDescr

Description:

Given the tolerance name, return general information for that tolerance. Further information on particular tolerance is given in either ExTol on InTol, depending on whether tolerance is specified as intrinsic or extrinsic.

Example file name: tol_clp

Retrieved from: tolerance

Key fields:
Tolerance_Name : integer

Data fields
Material_Condition : string {MMC, LMC, RFS}
{Note: if none, hold place with '-'}
Feature_Name : integer {feature you are tolerancing}
Expansion : Don't know -- hold place right now with '?'
Tolerance_Type : (IN, EX) {intrinsic or extrinsic}

16. ProbeChar

Description:
Given Probe_Characteristics_Name from InspectionPlan, return the relevant probe data.

Example file name: pbch_clp

Retrieved from: points

Key field:
Probe_Characteristics_Name : integer

Data fields:
Probe_Tip_Index : integer
Free_Space_Speed, Touch_Distance,
Touch_Speed, Retract_Distance, Over_Travel_Distance : real

17. SafePos

Description:
Specifies surface and point coordinates, given the safe point name.

Example file name: safe_clp

Retrieved from: cmm_tol

Key field:
Safe_Point_Name : integer

Data fields:
Surface_Name : integer {fake surface -- i.e. 99}
{Note: Point number for safe surface is always 0}
three components of point (x, y, z) : real

18. OneRowOf4x4

Description:
Returns one row (four components) of the 4x4 transformation matrix which relates the part coordinates to the machine coordinates.

Example file name: xrow_clp

Retrieved from: cmm_tol

Key fields:
Transformation_Name : integer
Row_Number : integer {1..4}

Data fields:
four components of one row of transformation : real
{4 real numbers}

19. Tols

Description:
Given the Tols_Name (from the InspectionPlan), get the tolerance name to do. Use this to retrieve the tolerance information from TolDescr.

Example file name: tols_clp

Retrieved from: cmm_tol

Key field:
Tols_Name : integer

Data field:
Tolerance_Name : integer

20. InspectionPlan

Description:
Given the inspection plan, return relevant inspection data (and pointers).

Example file name: plan_clp

Retrieved from: cmm_tol

```
Key fields:
       Inspection_Plan_Name : integer

Data fields:
       Units : (I, M) {measurement units for data -- inches or mm}
       Free_Space_Speed : real
       Transformation_Name, Safe_Name,
       Tols_Name,     {index to Tols relation--specifies tolerance
                      to do}
       Num_Of_Tols : integer
                 {use Tolerance_Name of 0 to specify intrinsic
                 tolerance to use to locate part}
```

21. SurfChars

```
Description:
       specifies the parameters for a particular surface

Example file name:    surf_clp

Retrieved from:       surfaces

Key fields:
       Surface_Name : integer

Data fields:
       NumOfPoints : integer
       SafeSurface    : (0 = non-safe,  1 = safe)
       SurfaceType : integer -- 1 --> plane
                                2 --> inside cylinder
                                3 --> outside cylinder
                                4 --> inside sphere
                                5 --> outside sphere
                                6 --> inside cone
                                7 --> outside cone
       SurfaceParms (Surface parameters -- fields 6 through 13.
       Note: data not required for all fields -- depends on
       surface type.)
            Data fields contain:

       * For a plane:       the X, Y, Z components of the
                            surface normal for the plane;  the
                            distance from the origin to the plane

            Example: 0.0, 0.0, 1.0, 1.87

       * For a cylinder:    the X, Y, Z coordinates of an arbitrary
                            point on the cylinder axis;  the X, Y,
```

Z components of a unit vector along the cylinder axis; the cylinder radius

Example: 0.938, 0.0, 0,938, 0.0, 1.0, 0.0, 0.406

* For a sphere: the X, Y, Z coordinates of the sphere center; the sphere radius

Example: 1.03, 1.50, 0.938, 0.250

* For a cone: the X, Y, Z coordinates of the cone vertex; the X, Y, Z components of a unit vector along the cone axis; the angle in radians of the cone half -- angle

Example: 1.938, 2.25, 0.333, 0.0, 1.0, 0.0, 0.933

READER COMMENT FORM

IMPLEMENTATION OF THE COORDINATE MEASURING MACHINE CONTROLLER

This document is one in a series of publications which document research done at the National Institute of Standards and Technology's Automated Manufacturing Research Facility from 1981 through March, 1987.

You may use this form to comment on the technical content or organization of this document or to contribute suggested editorial changes.

Comments: _____

_____

_____

_____

_____

_____

_____

_____

_____

If you wish a reply, give your name, company, and complete mailing address: _____

_____

_____

_____

What is your occupation? _____

NOTE: This form may not be used to order additional copies of this document or other documents in the series. Copies of AMRF documents are available from NTIS.

Please mail your comments to:

AMRF Program Manager
National Institute of
Standards and Technology
Building 220, Room B-111
Gaithersburg, MD 20899

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NISTIR 88-3874	2. Performing Organ. Report No.	3. Publication Date OCTOBER 1988

4. TITLE AND SUBTITLE

IMPLEMENTATION OF THE CMM CONTROLLER

5. AUTHOR(S)

Howard M. Moncarz, Theodore H. Hopp, Patrick A. Lezark

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

National Institute of Standards and Technology
Gaithersburg, Maryland 20899

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

This document describes the theory and implementation of the Coordinate Measuring Machine Controller (CMMC) program. This controller is part of the Inspection Workstation (IWS) in the Automated Manufacturing Research Facility (AMRF) at the National Institute of Standards and Technology. The CMMC supervises the coordinate measuring machine to perform a dimensional inspection of a part. This inspection is completely data-driven. The data consist of geometry, topology and tolerance information for the part as well as specific inspection data such as probe point sequencing information and probe characteristics.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

AMRF; CMM; coordinate measuring machine; data-driven control; dimensional inspection; inspection workstation; IWS

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

61

15. Price

$13.95